# Titre

Distribuer son code Python

# Contenu validé

# Oui

# Catch phrase (2 lignes max)

Découvrez comment utiliser correctement les modules de l’écosystème Python et les techniques pour diffuser vos propres bibliothèques. .

# Description (10 lignes max)

Le packaging de bibliothèques Python est l’un des domaines les plus mystérieux de ce langage. Pendant longtemps, ce domaine est resté le parent pauvre du langage, réservé à une communauté d’initiés. Les choses s’améliorent en ce moment.

Dans cette formation, vous verrez, dans un premier temps, comment profiter de l’écosystème des bibliothèques Python, nous verrons comment installer correctement des bibliothèques et rendre vos installations facilement duplicables . Dans un second temps, vous découvrirez comment contribuer à l’écosystème en ajoutant vos propres bibliothèques.

# 

# Identifiant technique

pysharing

# Domaine

développement

# Sous-domaine

Python

# Url

/formations/developpement/python-partager-son-code-python

# Ordre dans la page

7

# Prochaines sessions

## Lyon

21-22 juin 2021

## Lille

14-15 juin 2021

## Paris

28-29 juin 2021

# Équilibre théorie / pratique

35% théorie / 65% pratique

# Public visé / participants

* Développeurs
* Ingénieurs
* chefs de projets proches du développement.

# Prérequis

Bonnes maitrise de Python (syntaxe, fonctions, orienté objet, notion de modules et d’imports)

# Durée

2j - 14 heures

# Tarifs

1400 € HT par stagiaire (pauses et repas inclus)

# Objectifs pédagogiques

* Identifier et appliquer les bonnes pratiques d’installation de dépendances sur un projet Python
* Packager et déployer ses bibliothèques

# Plan de formation / programme

## **Rappels sur la gestion de bibliothèques Python**

### Les mécanismes d’import de code en Python et les chemins utilisés

* Qu’est-ce qu’un module python?
* Le cas de la bibliothèque standard
* L’installation de modules tiers (avec pip ou un gestionnaire de paquet Linux) et le dossier site-packages
* Importer du code et des modules
* Le PYTHONPATH et sa manipulation (au niveau de l’OS ou en Python directement)

### Les différentes façons de partager du code Python

* Les sources
* Les wheels
* Les autres gestionnaires de paquets

### Les environnements virtuels pour gérer facilement les projets

* Le concept et les cas d’usages des environnements virtuels
* Création et évolution d’environnements virtuels
* La notion de requirements.txt

### Le cas des bibliothèques complexes

* Reproduire une configuration de compilation
* Rôle de Docker, Vagrant, Ansible par rapport aux virtualenv

TP : importer et installer des bibliothèques, même anciennes que l’on doit compiler localement. Utilisation des fichiers de requirements pour réinstaller un projet et des environnements virtuels pour passer d’un projet à un autre.

## **Création de bibliothèques pour contribuer à l’écosystème Python**

### Rappel sur PyPI et pip

### La structure d’un module python

* Arborescence standard
* Les fichiers sources
* Les fichiers de tests
* Les autres fichiers

### Les fichiers de description du paquet

* Setup.py
* Setup.cfg
* L’ajout de fichiers tiers (données…)
* La création d’interface en ligne de commande (avec les entry points)

### La création proprement dite du paquet (source et wheel) et sa diffusion sur PyPI publique ou en privé (sur un dépôt git)

### Comment intégrer du code natif à un projet

* Les étapes de la compilation
* La création de wheels pour les différentes architectures

### Les autres moyens d’empaqueter (conda…)

TP : création d’un paquet simple et mise à disposition sur le dépôt de tests de PyPI et sur un git public

## Création et partage de programmes autonomes

### Présentation de l’outil PyInstaller

* Gestion des bibliothèques
* Gestion des fichiers tiers
* Utilisation sous Windows, Linux et MacOs

### Présentation de InnoSetup

TP : création d’un exécutable à partir d’un programme de démonstration et installation sur un poste n’ayant pas Python